FUTURE COMMANDS PLANNED FOR JLO SAFE V2

This note will detail what is currently planned to complete JLO SAFE V2 and explain a few ideas concerning SAFE V2.

The JLO SAFE V2 eprom is now very close to being completely full. When adding the last several commands, considerable attention was given to code space required by the new commands. I have been able to squeeze the last few commands out of very little eprom space, so I now feel I can still fit the following within the spare space still left on the eprom:

DISK NAME CHANGING. There have been many requests for the ability to change the name of a disk. I will implement a command to do this. Its syntax will probably be: RESTORE /"newdiskname". The fact that the token "TO" will not follow the new disk name can tell SAFE that we wish to change the disk name itself rather than a filename. This is the way the command will probably be done unless somebody suggests something better.

ERASE. An erase command for SAFE will be too long and complicated to fit on the SAFE V2 eprom itself, so I plan on the ERASE command to load a CODE (BYTES) file from disk into the Disk Board 'B' ram buffer. The code that will actually do the erase will be on a disk. Because of the way SAFE is structured, the erase command will take some time to remove a file if the file is physically located towards the beginning of a fairly full disk. This is because the command will reclaim disk space by physically "reshuffling" the files on the disk to close the gap left by the file. Erasing files towards the end of the disk (last few listed in a CAT) should be reasonably quick. I will recommend that only disks that are backed up have files erased, because if a power glich, etc., should occur during this reclaiming process, the entire disk is likely to be corrupted. Unlike MOST erase commands, however, the disk will NOT be cluttered by "gaps" or "holes" left by erasing a file. When the erase command is done, the disk will be as if the particular file had never existed in the first place. The use of a separate "compacting" program will never be needed!

With the addition of the diskname change and erase command, SAFE V2 will be in its final version, unless someone suggests something really desireable that would need little eprom code space to implement. At that time I plan on revising the Oliger Disk I/F Manual into its final form and offer a new copy to whomever would like one at a modest price. Then we can get on with some new HARDWARE projects!

Abbydale Designer's SPDOS is now available for the Oliger Disk I/F, as was previously used on the Ramex 2068 Disk I/F. This DOS is available on SAFE disk and contains all the commands of any other Abbydale SPDOS version. Its advantages are the many and varied commands available for use (a LOT!), its speed (very close speed-wise to SAFE V2), the fact that you can load files saved using a Ramex Disk I/F, and the fact that you can use SPDOS and SAFE V2 alternately within the same program by simply using the commands either would expect. Its disadvantages are the fact that it is ram based, so will use up about 4K of your computer's memory space and it will only work on your 2068 in its native 2068 Basic mode, not in Spectrum mode (the same restrictions as the Ramex I/F itself). Its price is $24.95 + $1.50 for post/pack and is available from: Cuyahoga Valley Software Works 615 School Ave. Cuyahoga Falls, OH 44221. Specify when ordering the 40 track or 80 track 5 1/4" version you require. These guys must pay a relativly high per piece royalty to Abbydale, are throwing in a MINIDOS on the disk that uses less ram than the regular SPDOS, and have updated the manual and are providing it with all orders. They will be making very little $$ for their efforts in providing this for you. If you can afford to throw in a few extra dollars as a "thank you" for all the time they have spent on this project, I am sure it will be appreciated!

Getting back to SAFE commands, I do NOT plan on adding a so-called sequential file to SAFE that is used via OPEN#, PRINT#, etc. In reality, all files used by SAFE ARE sequential files. The command most people are referring to when talking about a sequential file is really just another way of saving or loading text or numbers that must be done on lesser computers such as a Commodore or Atari. They are used on these computers because these machines lack the varied file types that SAFE and even 2068 Basic has built into its cassette handler, such as variables save with program, a variables only save (SAFE only), and numeric or character array save/loads. On a lesser computer if you dimension a character array you cannot directly save the array to disk when full of text, but must instead open a so-called sequential file and get within a loop to print it completely via print# to store it on disk. Sinclair users do not need to do this in such a crude way. We can simply save or load the array itself and be done. In summary, I am saying that this type of sequential file will not be implemented in SAFE V2 because it is simply not needed.

We can do it a better way!

A disassembly of SAFE V2.32 w/names (V2.40 soon, hopefully) is now available for $2.95pp. It also contains a sheet on accessing selected SAFE V2 functions from a machine code level.

I hope the information given here shows that I DO plan on continuing with SAFE V2 until I feel it is as complete as is required by its users. I wish to also make it clear that John Oliger Co. WILL be around to support this system and DOS for MANY years to come. You will NOT see Oliger CO. closing its doors to try and make a few faster $$ with a different computer brand. I very much like the 2068 computer and plan on supporting it as long as there is a need to support it. Yes, I bought a QL, but actually feel the 2068 is a far superior critter!

Jhn L. Oliger 10/87

PROBLEMS IN USING THE NEW V2.4 "MERGE" COMMAND

It has come to my attention that several JLO SAFE users have been having some trouble using the new SAFE "MERGE" command under certain conditions. This note will detail the results of my investigation into this problem, as requested by several Oliger Disk System owners.

I have had reports from a few users that indicated a problem may exist in SAFE causing the MERGE command to sometimes not work. The users reported that SAFE would act like it had MERGED the Basic code, return with an "OK" report, but when they would LIST to look at the additional code it would simply not be there. Repeated attempts to MERGE the BASIC into the same program would give the same result. However, when asked for a sample disk that contained the programs that resisted MERGE, nobody could provide me with the sample for one reason or another. Repeated attempts to cause the problem myself would always fail. I couldn't get it NOT to work!

Finally one customer called with a report of the problem, and he had it doing it right then, before our very eyes (and ears)! He told me he would make a copy of the disk having the problem and get it off to me to work with.

I recieved the disk and quickly went to work. The main program on the disk contained Bill Jone's (are you listening Bill?) "SMART DAISY" Basic program, and the Basic code to MERGE was a ten line or so eprom programming subroutine that started at line 9980. The last line in SMART DAISY was line 9520. I loaded the main program and then tried to merge the programming subroutine. As reported, the drive and SAFE ACTED like it had been merged but I sure couldn't see it there!

Further investigation found that I could not get a good merge after Smart Daisy was loaded even after the CLEAR command was used and even when Smart Daisy itself was removed using the DELETE , command!!! I found that <PRINT FREE> would return only 38289 bytes free after SMART DAISY had been loaded and then removed with CLEAR and DELETE , but on power up before any loads the computer had 38652 bytes free. I had somehow LOST 363 bytes of memory by loading SMART DAISY and then getting rid of it!

After running HOT Z AROS to find out just what was going on, I found that the pointers to the variables area and program area were not the same as they are when the computer is first powered up. Looking at where the computer thought the program was showed a lot of gibberish and portions of one of the screen copy routines used on the cassette supplied with the Oliger Printer Interface.

I checked out several copies of Bill Jone's SMART TEXT and found that EVERY ONE of them from the very first was missing those same 363 bytes. They ALL had the same gibberish where there should have been program lines at the end of the basic.

Evidently somewhere along the line (very early) Bill's program had been corrupted, either through glich, cassette data error, stray POKE(s), or whatever. The MERGED code is actually there, but Basic can't find it to LIST or RUN because the gibberish before it in memory confuses it.

Although all of Bill's Smart Text series programs seem to have this same corruption, there are likely to be other programs found that are also like this. Because of the way Basic programs are stored, saved, and loaded on the 2068, once something like this occurs and the program re-saved (using either cassette or disk), it will forever exist because as far as the computer is concerned it IS a part of the Basic program, even though it can't use it or LIST it.

I hope I haven't bored everyone to sleep by now. I've attempted to go through the entire process of how I try to resolve a problem like this reported to me. Sometimes I beat my head against the wall all day long and never get a problem solved. If you have a real strange problem with a certain command or function in SAFE just some of the time, PLEASE put it on disk as a sample and send it with your letter. It can REALLY be a big help!

So, how do you know if a program has this problem? If you cannot use the SAFE MERGE command with it then it is very likely. But the REAL test is to use the command <PRINT FREE> immediately after power up and again after the program is loaded and the removed using <CLEAR :DELETE ,>. If the numbers don't match you can be fairly certain that the program does contain corruption (unless it had self-run and used the CLEAR N command). I will now detail step by step how to fix this:

1) FIND AMOUNT OF PROGRAM MEMORY CORRUPTED:

A) Write down result of the statement <PRINT FREE> immediately after computer power-up. (38652 on standard 2068 w/o any cartridges installed)

B) Load corrupted program. Save variables to disk with the command <SAVE /"vars" VAL>. Clear both program and variables from the computer with the command line <CLEAR :DELETE ,>. Now write down result of the statement <PRINT FREE>. (38289 for Smart Text)

C) Subtract the number written down in 1B above from the number written down in 1A. Write this number down. (363 for Smart Text) This is the amount of memory corrupted.

2) FIND CORRECTED BASIC VARIABLES POINTER:

A) Re-load corrupted program and remove the variables again with the <CLEAR> command.

B) Enter the command <PRINT PEEK 23627+ PEEK 23628*256>. Write down this number.

C) Subtract the number obtained in 1C from the number you just wrote down in 2B. Write down the result.

3) FIND NEW VALUES TO POKE:

A) Divide the number obtained in 2C by 256. Round down to the next whole number and write this down. (MSB)

B) Multiply the number just written down in 3A by 256. Subtract this product from the number obtained in 2C. Write down the result. (LSB)

4) SET THE VARIABLES POINTER CORECTLY AND CLEAR THE CORRUPTION:

A) Set the variables pointer with the command <POKE 23627,LSB: POKE 23628,MSB> with "LSB" and "MSB" replaced by the numbers written down in 2B and 2A respectively.

B) Clear the corrupted memory by using the <CLEAR> command again.

5) GET IT ALL BACK ON DISK:

A) Re-load back in the old variables with the command <LOAD /"vars" VAL>.

B) Re-save the repaired file with the regular <SAVE /"Filename"> command.

The "cleaned-up" program should now allow MERGE to operate and be shorter to boot!

John L. Oliger 2/88